Where did my setup.py go? Changes in Python Recipes in Yocto Project 4.0 'kirkstone' Release

Tim Orling, Konsulko Group
Ross Burton, Arm

Yocto Project Summit *Virtual*, 2022.05

Konsulko Group

- Services company specializing in Embedded Linux and Open Source Software
- Hardware/software build, design, development, and training services
- Based in San Jose, CA with an engineering presence worldwide
- https://konsulko.com/

Agenda

- **Deprecated distutils**
- **PEP-517, wheels, pyproject.toml**
- **New PEP-517 classes**
- **Rust Extensions for Python**
  - setuptools_rust, pyo3 and python3-cryptography

```
commit 25c3c69bc77fbfca5c1162e264dcf1d2970df5d4
Author: Ross Burton <ross@openedhand.com>
Date:   Fri Sep 9 16:11:54 2005 +0000

Update for new upstream. I am now l33t OE hax0r\!
```

```
commit ed634d36d64af846b68424d38eaa86b464584840
Author: Tim Orling <ticotimo@gmail.com>
Date:   Tue Jul 15 16:32:33 2014 -0700

meta-python: create layer
```

Contents

Page Source (GitHub)

PEP 517 – A build-system independent format for source trees

Author	Nathaniel J. Smith <njs at pobox.com>, Thomas Kluyver <thomas at kluyver.me.uk>
BDFL-Delegate	Nick Coghlan <ncoghlan at gmail.com>

Discussions-To	Distutils-SIG list
Status	Final
Type	Standards Track
Created	30-Sep-2015
Post-History	01-Oct-2015, 25-Oct-2015, 19-May-2017, 11-Sep-2017
Resolution	Distutils-SIG message

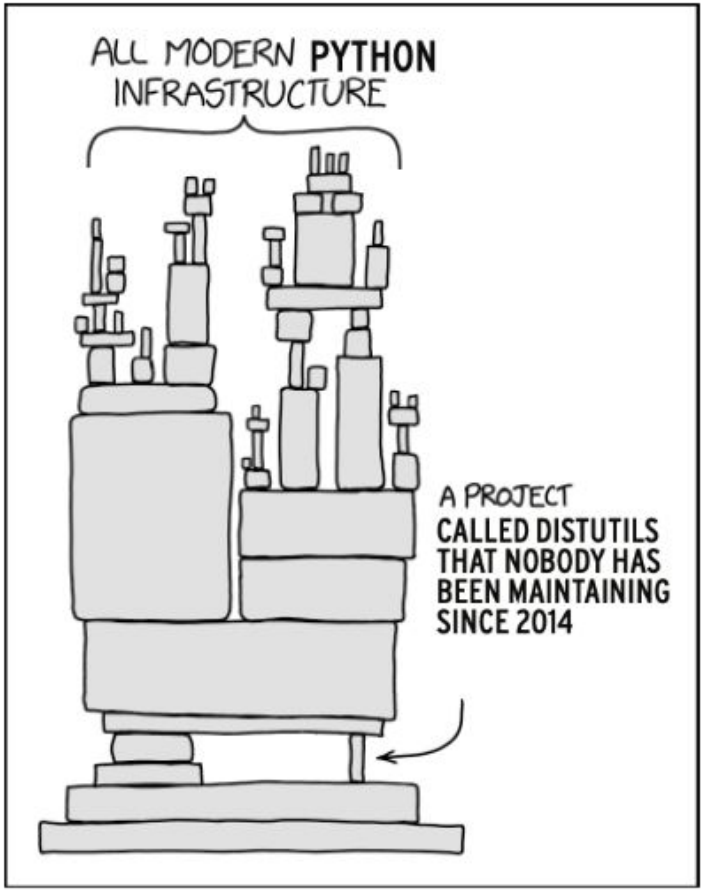
▶ **Table of Contents**

Abstract

While `distutils` / `setuptools` have taken us a long way, they suffer from three serious problems: (a) they're missing important features like usable build-time dependency declaration, autoconfiguration, and even basic ergonomic niceties like DRY-compliant version number management, and (b) extending them is difficult, so while there do exist various solutions to the above problems, they're often quirky, fragile, and expensive to maintain, and yet (c) it's very difficult to use anything else, because distutils/setuptools provide the standard interface for installing packages expected by both users and installation tools like `pip`.

Previous efforts (e.g. distutils2 or setuptools itself) have attempted to solve problems (a) and/or (b). This proposal aims to solve (c).

The goal of this PEP is get distutils-sig out of the business of being a gatekeeper for Python build systems. If you want to use distutils, great; if you want to use something else, then that should be easy to do using standardized methods. The difficulty of interfacing with distutils means that there aren't many such systems right now, but to give a sense of what we're thinking about see flit or bento. Fortunately, wheels have now solved many of the hard problems here – e.g. it's no longer necessary that a build system also know about every possible installation configuration – so pretty much all we really need from a build system is that it have some way to spit out standard-compliant wheels and sdists.



Thanks (and apologies) to https://xkcd.com/2347/

Contents

Page Source (GitHub)

PEP 632 – Deprecate distutils module

Author	Steve Dower <steve.dower at python.org>
Discussions-To	Discourse thread
Status	Accepted
Type	Standards Track
Created	03-Sep-2020
Python-Version	3.10
Post-History	03-Sep-2020, 22-Jan-2021
Resolution	Python-Dev thread

▶ **Table of Contents**

Abstract

The distutils module [1] has for a long time recommended using the setuptools package [2] instead. Setuptools has recently integrated a complete copy of distutils and is no longer dependent on the standard library [3]. Pip has been silently replacing distutils with setuptools when installing packages for a long time already, and the distutils documentation has stated that it is being phased out since 2014 (or earlier). It is time to remove it from the standard library.

Motivation

distutils [1] is a largely undocumented and unmaintained collection of utilities for packaging and distributing Python packages, including compilation of native extension modules. It defines a configuration format that describes a Python distribution and provides the tools to convert a directory of source code into a source distribution, and some forms of binary distribution. Because of its place in the standard library, many updates can only be released with a major release, and users cannot rely on particular fixes being available.

setuptools [2] is a better documented and well maintained enhancement based on distutils. While it provides very similar functionality, it is much better able to support users on earlier Python releases, and can respond to bug reports more quickly. A number of platform-specific enhancements already exist in setuptools that have not been added to distutils, and there is been a long-standing recommendation in the distutils documentation to prefer setuptools.

Porting from `distutils` to `setuptools`

```
-from distutils.core import setup
+from setuptools     import setup
```

PEP-517, pyproject.toml, wheels

```
$ cat pyproject.toml

[build-system]

requires = ["flit_core"]

build-backend = "flit_core.build_api"
```

```
def build_wheel(wheel_directory,
                config_settings=None,
                metadata_directory=None):

def build_sdist(sdist_directory,
                config_settings=None):
```

Contents

PEP 427 – The Wheel Binary Package Format 1.0

Author	Daniel Holth <dholth at gmail.com>
BDFL-Delegate	Nick Coghlan <ncoghlan at gmail.com>
Discussions-To	Distutils-SIG list
Status	Final
Type	Standards Track
Created	20-Sep-2012
Post-History	18-Oct-2012, 15-Feb-2013
Resolution	Python-Dev message

▶ **Table of Contents**

Canonical specification

The canonical version of the wheel format specification is now maintained at https://packaging.python.org/specifications/binary-distribution-format/ . This may contain amendments relative to this PEP.

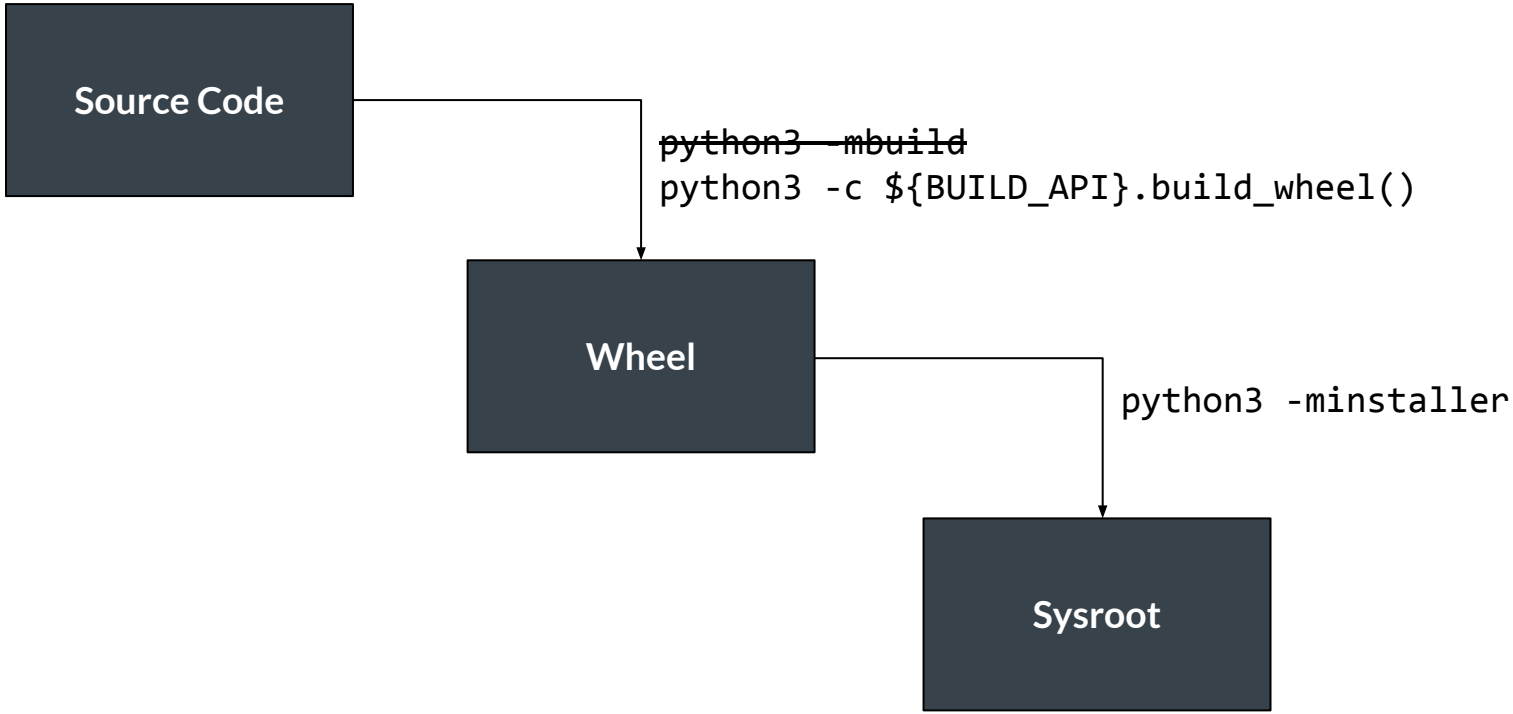
Abstract

This PEP describes a built-package format for Python called "wheel".

A wheel is a ZIP-format archive with a specially formatted file name and the `.whl` extension. It contains a single distribution nearly as it would be installed according to PEP 376 with a particular installation scheme. Although a specialized installer is recommended, a wheel file may be installed by simply unpacking into site-packages with the standard 'unzip' tool while preserving enough information to spread its contents out onto their final paths at any later time.

PEP Acceptance

This PEP was accepted, and the defined wheel version updated to 1.0, by Nick Coghlan on 16th February, 2013 [1]


Source Code
~~python3 -mbuild~~
python3 -c ${BUILD_API}.build_wheel()
Wheel
python3 -minstaller
Sysroot

New PEP-517 classes

when and how to use them

python_pep517.bbclass (abridged)

```
python_pep517_do_compile () {
    nativepython3 -c "import ${PEP517_BUILD_API} as api;
                      api.build_wheel('${PEP517_WHEEL_PATH}')"
}

python_pep517_do_install () {
    nativepython3 -m installer
                  --destdir=${D}
                  ${PEP517_WHEEL_PATH}/*.whl
}
```

```
$ cat pyproject.toml

[build-system]

requires = ["flit_core"]

build-backend = "flit_core.build_api"
```

Current PEP-517 Classes

- Flit Core: `python_flit_core.bbclass`
- Poetry Core: `python_poetry_core.bbclass`
- (Modern) Setuptools: `python_setuptools_build_meta.bbclass`

Porting from `setuptools` to `flit_core`

```
-inherit pypi setuptools3
+inherit pypi python_flit_core
```

Further Reading

- All of the referenced PEPs. Not for the faint of heart!
- *The State of Python Packaging*
  https://bernat.tech/posts/pep-517-and-python-packaging/

Rust Extensions for Python

setuptools_rust, pyo3 and python3-cryptography

Rust Extensions for Python

- https://pyo3.rs/
- https://github.com/pyo3
- https://github.com/PyO3/setuptools-rust

Setuptools plugin for Rust extensions

CI passing | pypi package 1.3.0 | docs passing | code style black

`setuptools-rust` is a plugin for `setuptools` to build Rust Python extensions implemented with PyO3 or rust-cpython.

Compile and distribute Python extensions written in Rust as easily as if they were written in C.

python_pyo3.bbclass (abridged)

```
inherit cargo python3-dir siteinfo

export PYO3_CROSS="1"
export PYO3_CROSS_PYTHON_VERSION="${PYTHON_BASEVERSION}"
export PYO3_CROSS_LIB_DIR="${STAGING_LIBDIR}"
export CARGO_BUILD_TARGET="${HOST_SYS}"
export RUSTFLAGS
export PYO3_PYTHON="${PYTHON}"
export PYO3_CONFIG_FILE="${WORKDIR}/pyo3.config"

python_pyo3_do_configure () {
    cat > ${WORKDIR}/pyo3.config << EOF
…}
```

python_setuptools_rust.bbclass

```
inherit python_pyo3 setuptools3

DEPENDS += "python3-setuptools-rust-native"

python_setuptools3_rust_do_configure() {
    python_pyo3_do_configure
    cargo_common_do_configure
    setuptools3_do_configure
}

EXPORT_FUNCTIONS do_configure
```

python3-cryptography_37.0.1.bb (abridged)

```
SRC_URI[sha256sum] = "..."

SRC_URI += "file://run-ptest \
           file://check-memfree.py \
           ...
           crate://crates.io/Inflector/0.11.4 \
           crate://crates.io/aliasable/0.1.3 \
           crate://crates.io/asn1/0.8.7 \
           ...
           crate://crates.io/winapi/0.3.9 \
           "

inherit pypi python_setuptools3_rust
```

Extracted from recipe generated with cargo-bitbake

https://github.com/meta-rust/cargo-bitbake

examples

```
Some recipe example of old and new
```

Next steps

Next steps

- recipetool and devtools aware of the new classes and recipes styles
- Maturin for Rust centric applications
  - https://github.com/pyo3/maturin
  - https://maturin.rs/

Thank You

- Everyone that helped get these changes across the finish line
- Richard Purdie for taking the changes at a late stage and doing an immense amount of work to help stabilize oe-core
- Tim especially thanks Ross Burton for the second set of eyes and all the refactoring

Happy Trails!

yocto PROJECT | THE LINUX FOUNDATION

What is the Yocto Project® ?

IT'S NOT AN EMBEDDED LINUX DISTRIBUTION, IT CREATES A CUSTOM ONE FOR YOU.

The Yocto Project (YP) is an open source collaboration project that helps developers create custom Linux-based systems regardless of the hardware architecture.

The project provides a flexible set of tools and a space where embedded developers worldwide can share technologies, software stacks, configurations, and best practices that can be used to create tailored Linux images for embedded and IOT devices, or anywhere a customized Linux OS is needed.

yocto PROJECT | THE LINUX FOUNDATION